Categorical Semantics for Logic-Enriched Type Theories

Robin Adams

Royal Holloway, University of London

TYPES 2011, 8 September 2011
The Curry-Howard Isomorphism

There are two facts that are both sometimes referred to as the Curry-Howard isomorphism. One is trivial, one is not.

Non-trivial Fact
When we do so:
the rules for conjunction are the rules for product type;
the rules for implication are the rules for non-dependent function type;
the rules for universal quantification are (almost) the rules for dependent function type;
the rules for classical logic are the rules for control operators (usually);
the rules for modal logic are the rules for metavariables;
etc.

In this talk, 'Curry-Howard' shall mean the second.
The Curry-Howard Isomorphism

There are two facts that are both sometimes referred to as the Curry-Howard isomorphism. One is trivial, one is not.

### Trivial Fact

It is possible to write a linear syntax for natural deduction proofs, and then write $\Gamma \vdash P : \phi$ for ‘$P$ is a proof of $\phi$ (that depends on the free variables and hypotheses $\Gamma$)’
The Curry-Howard Isomorphism

There are two facts that are both sometimes referred to as the Curry-Howard isomorphism. One is trivial, one is not.

Non-trivial Fact

When we do so:

- the rules for conjunction are the rules for product type;
- the rules for implication are the rules for non-dependent function type;
- the rules for universal quantification are (almost) the rules for dependent function type;
- the rules for classical logic are the rules for control operators (usually);
- the rules for modal logic are the rules for metavariables;
- etc.
The Curry-Howard Isomorphism

There are two facts that are both sometimes referred to as the *Curry-Howard isomorphism*. One is trivial, one is not.

### Non-trivial Fact

When we do so:

- the rules for conjunction are the rules for product type;
- the rules for implication are the rules for non-dependent function type;
- the rules for universal quantification are (almost) the rules for dependent function type;
- the rules for classical logic are the rules for control operators (usually);
- the rules for modal logic are the rules for metavariables;
- etc.

In this talk, ‘Curry-Howard’ shall mean the second.
My Beliefs on Curry-Howard

I believe:

1. Curry-Howard is surprising,
2. There is something there to be explained. (Why do propositions behave like types?)
3. We do not have a good explanation yet. (Propositions are not literally types.)
4. We are having problems because we tacitly assume propositions-as-types.
5. We should instead turn Curry-Howard into a mathematical object.
My Beliefs on Curry-Howard

I believe:

1. Curry-Howard is surprising,
My Beliefs on Curry-Howard

I believe:

1. Curry-Howard is surprising,
2. There is something there to be explained. (Why do propositions behave like types?)
My Beliefs on Curry-Howard

I believe:

1. Curry-Howard is surprising,
2. There is something there to be explained. (Why do propositions behave like types?)
3. We do not have a good explanation yet. (Propositions are not literally types.)
My Beliefs on Curry-Howard

I believe:

1. Curry-Howard is surprising,
2. There is something there to be explained. (Why do propositions behave like types?)
3. We do not have a good explanation yet. (Propositions are not literally types.)
4. We are having problems because we tacitly assume propositions-as-types.
My Beliefs on Curry-Howard

I believe:

1. Curry-Howard is surprising,
2. There is something there to be explained. (Why do propositions behave like types?)
3. We do not have a good explanation yet. (Propositions are not literally types.)
4. We are having problems because we tacitly assume propositions-as-types.
5. We should instead turn Curry-Howard into a mathematical object.
Introduction

We have:

- systems of first-order arithmetic
Introduction

We have:

- systems of first-order arithmetic
- systems of second-order arithmetic

It is very difficult to translate between the systems on the left, and the systems on the right. Syntax and semantics are both very different.

Logic-enriched type theories (LTTs) help.
Introduction

We have:

- systems of first-order arithmetic
- systems of second-order arithmetic
- set theories

It is very difficult to translate between the systems on the left, and the systems on the right. Syntax and semantics are both very different.

Logic-enriched type theories (LTTs) help.

Robin Adams (RHUL)
Introduction

We have:

- systems of first-order arithmetic
- systems of second-order arithmetic
- set theories
- type theories

It is very difficult to translate between the systems on the left, and the systems on the right. Syntax and semantics are both very different.

Logic-enriched type theories (LTTs) help.
Introduction

We have:

- systems of first-order arithmetic
- systems of second-order arithmetic
- set theories
- type theories
- ...
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
It is very difficult to translate between the systems on the left, and the systems on the right.

Syntax and semantics are both very different. Logic-enriched type theories (LTTs) help.
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
  - If propositions really were types, it should be easy.
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and *semantics* are both very different.
It is very difficult to translate between the systems on the left, and the systems on the right.

Syntax and *semantics* are both very different.

*Logic-enriched type theories (LTTs)* help.
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and semantics are both very different.

- Logic-enriched type theories (LTTs) help.
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and semantics are both very different.
- Logic-enriched type theories (LTTs) help.
- Syntactic translations are possible.
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and *semantics* are both very different.
- *Logic-enriched type theories* (*LTTs*) help.
- Syntactic translations are possible.
- Curry-Howard becomes just one of a family.
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and semantics are both very different.

- *Logic-enriched type theories (LTTs)* help.

- Syntactic translations are not enough.
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and semantics are both very different.
- Logic-enriched type theories (LTTs) help.
- We need a semantics for LTTs.
1 Logic-Enriched Type Theories
   - Syntax

2 Categorical Semantics
   - Introduction to Categorical Semantics
   - Categorical Semantics for Logic-Enriched Type Theories
   - Soundness and Completeness Theorems

3 Applications
   - Conservativity of ACA₀ over PA
   - Bounded Quantification
Syntax of an LTT

$LTT_0$ is a system with:

Judgement forms:

\[
\begin{align*}
\Gamma &\vdash A \text{ Type} & \Gamma &\vdash M : A \\
\Gamma &\vdash \phi \text{ Prop} & \Gamma &\vdash P : \phi
\end{align*}
\]

and associated equality judgements.
Syntax of an LTT

$LTT_0$ is a system with:

- arrow types $A \rightarrow B$
  
  with objects $\lambda x : A. M$
Syntax of an LTT

$LTT_0$ is a system with:

- **arrow types** $A \rightarrow B$
- **product types** $A \times B$
  with objects $(M, M)$
Syntax of an LTT

$LTT_0$ is a system with:

- arrow types $A \to B$
- product types $A \times B$
- natural numbers $\mathbb{N}$
  with objects $0$ and $S(M)$
Syntax of an LTT

$LTT_0$ is a system with:

- arrow types $A \to B$
- product types $A \times B$
- natural numbers $\mathbb{N}$
- a type universe $U$
  with objects $\hat{\mathbb{N}}$ and $M \times M$
Syntax of an LTT

LTT$_0$ is a system with:

- arrow types $A \to B$
- product types $A \times B$
- natural numbers $\mathbb{N}$
- a type universe $U$
- classical predicate logic
  with propositions $M =_A M$, $\neg \phi$, $\phi \land \psi$, $\forall x : A.\phi$, ...
Syntax of an LTT

$LTT_0$ is a system with:

- arrow types $A \rightarrow B$
- product types $A \times B$
- natural numbers $\mathbb{N}$
- a type universe $U$
- classical predicate logic
- a *propositional universe* $\text{prop Type}$
Syntax of an LTT

$LTT_0$ is a system with:

- arrow types $A \rightarrow B$
- product types $A \times B$
- natural numbers $\mathbb{N}$
- a type universe $U$
- classical predicate logic
- a *propositional universe* $\text{prop } \text{Type}$

We write $\text{Set } (A)$ for $A \rightarrow \text{prop}$.
The Propositional Universe

A *type universe* is a type whose objects are names of types:
- The universe $U$ contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \times \hat{\mathbb{N}}$, . . . .

A *propositional universe* is a type whose objects are names of propositions:
- The universe $\text{prop}$ contains $\hat{\neg} \hat{0} \hat{=} \hat{\mathbb{N}} S(0)$, $\hat{\forall} x : \hat{\mathbb{N}}. x \hat{=} \hat{\mathbb{N}} x$, etc.

In $LTT_0$, $\text{prop}$ contains the propositions that do not involve quantification over large types.

A proposition is *small* iff it has a name in $\text{prop}$, *large* otherwise.

The strength of an LTT is determined by which types and which propositions we can eliminate over.

We can only eliminate $\hat{\mathbb{N}}$ over small types.

We can only use proof by induction with small propositions.

Adding a new type or connective is conservative. Adding it to the universes is not.
The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe $U$ contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \times \hat{\mathbb{N}}$, \ldots.
- A type is *small* iff it has a name in $U$, *large* otherwise.
The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe $U$ contains $\hat{\mathbb{N}}, \hat{\mathbb{N}} \times \hat{\mathbb{N}}, \ldots$.
- A type is *small* iff it has a name in $U$, *large* otherwise.
The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe $U$ contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \times \hat{\mathbb{N}}$, . . . .
- A type is *small* iff it has a name in $U$, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe $\text{prop}$ contains $\neg 0 \equiv_{\hat{\mathbb{N}}} S(0)$, $\forall x : \hat{\mathbb{N}}. x \equiv_{\hat{\mathbb{N}}} x$, etc.

We can only eliminate $\mathbb{N}$ over small types.
We can only use proof by induction with small propositions.
Adding a new type or connective is conservative. Adding it to the universes is not.
The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe $U$ contains $\mathbb{N}$, $\mathbb{N} \times \mathbb{N}$, $\ldots$.
- A type is *small* iff it has a name in $U$, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe $\text{prop}$ contains $\neg 0 \equiv \mathbb{N} S(0)$, $\forall x : \mathbb{N}. x \equiv \mathbb{N} x$, etc.
- In $\text{LTT}_0$, $\text{prop}$ contains the propositions that do not involve quantification over large types.
The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe $U$ contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \times \hat{\mathbb{N}}$, . . . .
- A type is *small* iff it has a name in $U$, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe $\text{prop}$ contains $\hat{\neg}0 \hat{=} \hat{\mathbb{N}}S(0)$, $\forall x : \hat{\mathbb{N}}. x \hat{=} \hat{\mathbb{N}}x$, etc.
- In $\text{LTT}_0$, $\text{prop}$ contains the propositions that do not involve quantification over large types.
- A proposition is *small* iff it has a name in $\text{prop}$, *large* otherwise.
The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe $U$ contains $\hat{\mathbb{N}}, \hat{\mathbb{N}} \times \hat{\mathbb{N}}, \ldots$.
- A type is *small* iff it has a name in $U$, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe $\text{prop}$ contains $\hat{\neg}0 \hat{=} \hat{\mathbb{N}} S(0), \hat{\forall} x : \hat{\mathbb{N}}. x \hat{=} \hat{\mathbb{N}} x$, etc.
- In $\text{LTT}_0$, $\text{prop}$ contains the propositions that do not involve quantification over large types.
- A proposition is *small* iff it has a name in $\text{prop}$, *large* otherwise.
The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe $U$ contains $\hat{\mathbb{N}}, \hat{\mathbb{N}} \times \hat{\mathbb{N}}, \ldots$.
- A type is *small* iff it has a name in $U$, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe $\text{prop}$ contains $\neg 0 \equiv_{\mathbb{N}} S(0), \forall x : \hat{\mathbb{N}}. x \equiv_{\hat{\mathbb{N}}} x$, etc.
- In $\text{LTT}_0$, $\text{prop}$ contains the propositions that do not involve quantification over large types.
- A proposition is *small* iff it has a name in $\text{prop}$, *large* otherwise.

The strength of an LTT is determined by which types and which propositions we can *eliminate* over.

- We can only eliminate $\mathbb{N}$ over small types.
The Propositional Universe

A *type universe* is a type whose objects are names of types:
- The universe $U$ contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \times \hat{\mathbb{N}}$, $\ldots$.
- A type is *small* iff it has a name in $U$, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:
- The universe $\text{prop}$ contains $\neg 0 \equiv_{\hat{\mathbb{N}}} S(0)$, $\forall x : \hat{\mathbb{N}}. x \equiv_{\hat{\mathbb{N}}} x$, $\ldots$.
- In $\text{LTT}_0$, $\text{prop}$ contains the propositions that do not involve quantification over large types.
- A proposition is *small* iff it has a name in $\text{prop}$, *large* otherwise.

The strength of an LTT is determined by which types and which propositions we can *eliminate* over:
- We can only eliminate $\mathbb{N}$ over small types.
- We can only use proof by induction with small propositions.
The Propositional Universe

A type universe is a type whose objects are names of types:
- The universe $U$ contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \times \hat{\mathbb{N}}$, . . . .
- A type is small iff it has a name in $U$, large otherwise.

A propositional universe is a type whose objects are names of propositions:
- The universe $\text{prop}$ contains $\hat{\neg}0 \equiv_{\hat{\mathbb{N}}} S(0)$, $\forall x : \hat{\mathbb{N}}. x \equiv_{\hat{\mathbb{N}}} x$, etc.
- In LTT$_0$, prop contains the propositions that do not involve quantification over large types.
- A proposition is small iff it has a name in prop, large otherwise.

The strength of an LTT is determined by which types and which propositions we can eliminate over.
- We can only eliminate $\mathbb{N}$ over small types.
- We can only use proof by induction with small propositions.
- Adding a new type or connective is conservative. Adding it to the universes is not.
We can give semantics to a type theory in a variety of ways:

Map types to sets, $\omega$-sets, PERs, sheaves, domains, \ldots

To save repeating work, we:

- define the properties a category must have for us to build a semantics from its objects;
- give semantics to the theory in an *arbitrary* category with those properties.
Categorical Semantics for a Dependent Type Theory

To give semantics to a dependent type theory, we need:

- a category $\mathcal{B}$ (whose objects interpret contexts $\Gamma$);

Such that $p = \text{cod} \circ P$ is a fibration $\mathcal{B}$ has a terminal object...
To give semantics to a dependent type theory, we need:

- a category $\mathcal{B}$ (whose objects interpret contexts $\Gamma$);
- a category $\mathcal{E}$ (whose objects interpret types-in-context $\Gamma \vdash A \text{ Type}$);

such that $p = \text{cod} \circ P$ is a fibration $\mathcal{B}$ has a terminal object.
To give semantics to a dependent type theory, we need:

- a category $\mathcal{B}$ (whose objects interpret contexts $\Gamma$);
- a category $\mathcal{E}$ (whose objects interpret types-in-context $\Gamma \vdash A \text{ Type}$);
- a functor $\mathcal{P}: \mathcal{E} \to \mathcal{B}$ (mapping $\Gamma \vdash A$ to $(\Gamma, x : A) \to \Gamma$).

\[ \begin{array}{ccc}
\mathcal{E} & \xrightarrow{\mathcal{P}} & \mathcal{B} \\
\downarrow p & & \downarrow \text{cod} \\
\mathcal{B} & & 
\end{array} \]
To give semantics to a dependent type theory, we need:

- a category $\mathcal{B}$ (whose objects interpret contexts $\Gamma$);
- a category $\mathcal{E}$ (whose objects interpret types-in-context $\Gamma \vdash A \text{ Type}$);
- a functor $\mathcal{P} : \mathcal{E} \to \mathcal{B}$ → (mapping $\Gamma \vdash A$ to $(\Gamma, x : A) \to \Gamma$).

such that

- $p = \text{cod} \circ \mathcal{P}$ is a fibration
- $\mathcal{B}$ has a terminal object
Categorical Semantics for a LTT

To give semantics to a LTT, we need, in addition:

- a category $\mathcal{P}$ (whose objects interpret propositions-in-context $\Gamma \vdash \phi$ Prop);

\[
\begin{array}{ccc}
\mathcal{P} & \xrightarrow{\mathcal{P}} & B \\
E & \xrightarrow{p} & B \\
\end{array}
\]
Categorical Semantics for a LTT

To give semantics to a LTT, we need, in addition:

- a category $\mathcal{P}$ (whose objects interpret propositions-in-context $\Gamma \vdash \phi \text{ Prop}$);
- a fibration $q : \mathcal{P} \to \mathcal{B}$ (mapping $\Gamma \vdash \phi \text{ Prop}$ to $\Gamma$)

\[ \begin{align*}
\mathcal{P} & \xrightarrow{\varphi} \mathcal{E} \\
\mathcal{E} & \xrightarrow{\mathcal{P}} \mathcal{B} \\
\mathcal{B} & \xrightarrow{\text{cod}} \mathcal{B} \rightarrow
\end{align*} \]
Categorical Semantics for a LTT

To give semantics to a LTT, we need, in addition:

- a category $\mathcal{P}$ (whose objects interpret \textit{propositions-in-context} $\Gamma \vdash \phi \text{ Prop}$);
- a fibration $q : \mathcal{P} \to \mathcal{B}$ (mapping $\Gamma \vdash \phi \text{ Prop}$ to $\Gamma$)
- for every object $\Gamma \vdash A \text{ Type in } \mathcal{E}$, a right adjoint $\pi^* \dashv \forall$ and a left adjoint $\exists \dashv \pi^*$
Categorical Semantics for a LTT

To give semantics to a LTT, we need, in addition:

- a category $\mathcal{P}$ (whose objects interpret propositions-in-context $\Gamma \vdash \phi \text{ Prop}$);

- a fibration $q : \mathcal{P} \to \mathcal{B}$ (mapping $\Gamma \vdash \phi \text{ Prop}$ to $\Gamma$)

- for every object $\Gamma \vdash A \text{ Type}$ in $\mathcal{E}$, a right adjoint $\pi^* \dashv \forall$ and a left adjoint $\exists \dashv \pi^*$

- such that $\mathcal{P}$ is a locally Cartesian closed category.
Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe $(U, T)$, we need:

- a substructure (intended to represent the small types and small contexts)

\[
\begin{array}{ccc}
D & 
\rightarrow & E \\
\downarrow & & \downarrow \\
A & 
\rightarrow & B
\end{array}
\]
Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe \((U, T)\), we need:

- a substructure (intended to represent the small types and small contexts)
- an object \(U\) in \(\mathcal{E}\) over the empty context (terminal object)

We require \(\top \rightarrow \langle x : N \rangle \rightarrow \langle x : N \rangle\) to be a weak fibred natural number object in both of these right-hand-sides.
Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe \((U, T)\), we need:

- a substructure (intended to represent the small types and small contexts)
- an object \(U\) in \(E\) over the empty context (terminal object)
- a generic object \(x : U \vdash T(x)\ Type\ in E\) over \(\text{dom} \ P(\vdash U\ Type)\);

We require \(\top \rightarrow \langle x : N \rangle \rightarrow \langle x : N \rangle\) to be a weak fibred natural number object in both of these right-hand-sides.

\[
\begin{array}{ccc}
D & \xrightarrow{} & D \times_A B \\
\downarrow & & \downarrow \\
A & \xleftarrow{} & B
\end{array}
\]
To give semantics to a dependent type theory with a universe \((U, T)\), we need:

- a substructure (intended to represent the small types and small contexts)
- an object \(U\) in \(E\) over the empty context (terminal object)
- a generic object \(x : U \vdash T(x) \text{ Type}\) in \(E\) over \(\text{dom } \mathcal{P}(\vdash U \text{ Type})\);

To give semantics to \((\text{prop}, V)\), we need in addition:

- a substructure (intended to represent the small propositions and contexts consisting solely of small propositions)
- an object \(\text{prop}\) in \(E\) over the terminal object
- a generic object \(V\) in \(E\) over \(\text{dom } \mathcal{P}(\vdash \text{prop})\).

We require \(\top \to \langle x : N \rangle \to \langle x : N \rangle\) to be a weak fibred natural number object in both of these right-hand-sides.
Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe \((U, T)\), we need:

- a substructure (intended to represent the small types and small contexts)
- an object \(U\) in \(\mathcal{E}\) over the empty context (terminal object)
- a generic object \(x : U \vdash T(x)\) Type in \(\mathcal{E}\) over \(\text{dom} \ P(\vdash U\ \text{Type})\);

To give semantics to \((\text{prop}, V)\), we need in addition:

- a substructure (intended to represent the small propositions and contexts consisting solely of small propositions)
Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe \((U, T)\), we need:

- a substructure (intended to represent the small types and small contexts)
- an object \(U\) in \(E\) over the empty context (terminal object)
- a generic object \(x : U \vdash T(x)\ Type\) in \(E\) over \(\text{dom } P(\vdash U \ Type)\);

To give semantics to \((\text{prop}, V)\), we need in addition:

- a substructure (intended to represent the small propositions and contexts consisting solely of small propositions)
- an object \(\text{prop}\) in \(E\) over the terminal object
Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe \((U, T)\), we need:

- a substructure (intended to represent the small types and small contexts)
- an object \(U\) in \(\mathcal{E}\) over the empty context (terminal object)
- a generic object \(x : U \vdash T(x) \text{ Type}\) in \(\mathcal{E}\) over \(\text{dom} \mathcal{P}(\vdash U \text{ Type})\);

To give semantics to \((\text{prop}, V)\), we need in addition:

- a substructure (intended to represent the small propositions and contexts consisting solely of small propositions)
- an object \(\text{prop}\) in \(\mathcal{E}\) over the terminal object
- a generic object \(V\) in \(\mathcal{E}\) over \(\text{dom} \mathcal{P}_{\text{prop}}\).

\[
\begin{array}{c}
\mathcal{Q} \\
\downarrow \\
\mathcal{C}
\end{array}
\quad
\begin{array}{c}
\mathcal{Q} \times_{\mathcal{C}} \mathcal{B} \\
\downarrow \\
\mathcal{B}
\end{array}
\]
Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe \((U, T)\), we need:

- a substructure (intended to represent the small types and small contexts)
- an object \(U\) in \(\mathbb{E}\) over the empty context (terminal object)
- a generic object \(x : U \vdash T(x) \text{Type}\) in \(\mathbb{E}\) over \(\text{dom} \mathcal{P}(\vdash U \text{Type})\);

To give semantics to \((\text{prop}, V)\), we need in addition:

- a substructure (intended to represent the small propositions and contexts consisting solely of small propositions)
- an object \(\text{prop}\) in \(\mathbb{E}\) over the terminal object
- a generic object \(V\) in \(\mathbb{E}\) over \(\text{dom} \mathcal{P}_{\text{prop}}\).
Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe \((U, T)\), we need:

- a substructure (intended to represent the small types and small contexts)
- an object \(U\) in \(\mathbb{E}\) over the empty context (terminal object)
- a generic object \(x : U \vdash T(x) \text{ Type}\) in \(\mathbb{E}\) over \(\text{dom} \mathcal{P}(\vdash U \text{ Type})\);

To give semantics to \((\text{prop}, V)\), we need in addition:

- a substructure (intended to represent the small propositions and contexts consisting solely of small propositions)
- an object \(\text{prop}\) in \(\mathbb{E}\) over the terminal object
- a generic object \(V\) in \(\mathbb{E}\) over \(\text{dom} \mathcal{P}_{\text{prop}}\).

We require \(\top \rightarrow \langle x : \mathbb{N} \rangle \rightarrow \langle x : \mathbb{N} \rangle\) to be a weak fibred natural number object in both of these right-hand-sides.
Interpretation

Given an LTT\textsubscript{W}-category \( \mathcal{C} \), define:

- for every valid context \( \Gamma \), an object \( \llbracket \Gamma \rrbracket \) of \( \mathbb{B} \);
- for every type \( A \) such that \( \Gamma \vdash A \ Type \), an object \( \llbracket \Gamma \vdash A \rrbracket \) of \( \mathbb{E} \) such that \( p \llbracket \Gamma \vdash A \rrbracket = \llbracket \Gamma \rrbracket \)
- for every term \( M \) such that \( \Gamma \vdash M : A \), an arrow \( \llbracket \Gamma \vdash M \rrbracket : \llbracket \Gamma \rrbracket \to \text{dom} \mathcal{P} \llbracket \Gamma \vdash A \rrbracket \)
- for every proposition \( \phi \) such that \( \Gamma \vdash \phi \prop \), an object \( \llbracket \Gamma \vdash \phi \rrbracket \) of \( \mathbb{P} \) over \( \llbracket \Gamma \rrbracket \)
Soundness Theorem

Theorem

Every judgement is true in any LTT\(_W\)-category. That is:

1. If \(\Gamma \vdash A = B\) then \([\Gamma \vdash A] = [\Gamma \vdash B]\)
2. If \(\Gamma \vdash M = N : A\) then \([\Gamma \vdash M] = [\Gamma \vdash N]\)
3. If \(\Gamma \vdash \phi = \psi\) then \([\Gamma \vdash \phi] = [\Gamma \vdash \psi]\)
4. If there is a proof \(\Gamma \vdash P : \phi\) then there is a vertical arrow \(\top \to [\Gamma \vdash \phi]\) in the fibre \(P/\Gamma\).

Proof.

Induction on derivations.
Completeness Theorem

**Theorem**

If a judgement is true in every category $\mathcal{C}$, then it is derivable in $T$.

**Proof.**

Define the category $\text{Cl}(T)$, the *classifying category* of $T$, thus:

- the objects of $\mathbb{B}$ are the valid contexts;
- the objects of $\mathbb{E}$ are the pairs $(\Gamma, A)$ such that $\Gamma \vdash A \text{ Type}$, quotiented by equality;
- ... 

If a judgement is true in $\text{Cl}(T)$, then it is derivable in $T$.  

---

Robin Adams (RHUL)  
Categorical Semantics for LTTs  
TYPES 2011 14 / 21
Completeness Theorem

Theorem

If a judgement is true in every category $\mathcal{C}$, then it is derivable in $T$.

Proof.

Define the category $\text{Cl}(T)$, the *classifying category* of $T$, thus:

- the objects of $\mathbb{B}$ are the valid contexts;
- the objects of $\mathbb{E}$ are the pairs $(\Gamma, A)$ such that $\Gamma \vdash A$ Type, quotiented by equality;
- ...  

If a judgement is true in $\text{Cl}(T)$, then it is derivable in $T$.

In fact, $\text{Cl}(T)$ is an initial object in the metacategory of $\text{LTT}_W$-categories. The interpretation given earlier is the unique functor $\text{Cl}(T) \to \mathcal{C}$.
Completeness Theorem

**Theorem**

*If a judgement is true in every category $C$, then it is derivable in $T$.***

**Proof.**

Define the category $\text{Cl}(T)$, the *classifying category* of $T$, thus:

- the objects of $B$ are the valid contexts;
- the objects of $E$ are the pairs $(\Gamma, A)$ such that $\Gamma \vdash A$ Type, quotiented by equality;
- ...

If a judgement is true in $\text{Cl}(T)$, then it is derivable in $T$. □

In fact, $\text{Cl}(T)$ is an initial object in the metacategory of $\text{LTT}_W$-categories. The interpretation given earlier is the unique functor $\text{Cl}(T) \to C$. This is the sort of thing that gets category theorists excited.
Conservativity of $\text{LTT}_0$ over $\text{PA}$

I have previously given *syntactic* proofs that $\text{LTT}_0$ is conservative over $\text{PA}$. We can now give a *semantic* proof of the same result.

**Theorem**

$LTT_0$ is conservative over $\text{PA}$.

**Proof.**

From any model $\mathcal{M}$ of $\text{PA}$, we construct a model of $\text{LTT}_0$. 
Conservativity of $\text{LTT}_0$ over $\text{PA}$

I have previously given syntactic proofs that $\text{LTT}_0$ is conservative over $\text{PA}$. We can now give a semantic proof of the same result.

**Theorem**

$LTT_0$ is conservative over $\text{PA}$.

**Proof.**

From any model $\mathcal{M}$ of $\text{PA}$, we construct a model of $LTT_0$. Define the higher-order recursive (hor) functions to be those built up from $0^\mathcal{M}$ and $S^\mathcal{M}$ by composition, primitive recursion, pairing, projection, lambda-abstraction and application.
Conservativity of $\text{LTT}_0$ over PA

I have previously given syntactic proofs that $\text{LTT}_0$ is conservative over PA. We can now give a semantic proof of the same result.

**Theorem**

$LTT_0$ is conservative over PA.

**Proof.**

From any model $\mathcal{M}$ of PA, we construct a model of $\text{LTT}_0$. Define the higher-order recursive (hor) functions to be those built up from $0^\mathcal{M}$ and $S^\mathcal{M}$ by composition, primitive recursion, pairing, projection, lambda-abstraction and application. Define the arithmetic predicates to be those built up from equality by Boolean operations and quantification over $|\mathcal{M}|$. 

We can similarly prove $\text{LTT}_0$ conservative over $\text{ACA}_0$.

**Corollary**

$\text{ACA}_0$ is conservative over PA.
Conservativity of LTT$_0$ over PA

I have previously given syntactic proofs that LTT$_0$ is conservative over PA. We can now give a semantic proof of the same result.

**Theorem**

LTT$_0$ is conservative over PA.

**Proof.**

From any model $\mathcal{M}$ of PA, we construct a model of LTT$_0$. The objects of $\mathbb{E}$ are the sets built up from $|\mathcal{M}|$ by $\times$, $\rightarrow$ and $P$, where $A \rightarrow B$ is the set of hom functions from $A$ to $B$, and $PA$ is the set of arithmetic subsets of $A$. 

Robin Adams (RHUL)
Categorical Semantics for LTTs
TYPES 2011 15 / 21
Conservativity of LTT\(_0\) over PA

I have previously given *syntactic* proofs that LTT\(_0\) is conservative over PA. We can now give a *semantic* proof of the same result.

**Theorem**

LTT\(_0\) is conservative over PA.

**Proof.**

From any model \(\mathcal{M}\) of PA, we construct a model of LTT\(_0\). The objects of \(\mathcal{E}\) are the sets built up from \(|\mathcal{M}|\) by \(\times\), \(\rightarrow\) and \(P\). The objects of \(\mathcal{B}\) are the sets of all sequences of objects of \(\mathcal{E}\). The arrows are the hor functions.
Conservativity of $\text{LTT}_0$ over $\text{PA}$

I have previously given *syntactic* proofs that $\text{LTT}_0$ is conservative over $\text{PA}$. We can now give a *semantic* proof of the same result.

**Theorem**

$LTT_0$ *is conservative over* $\text{PA}$.

**Proof.**

From any model $\mathcal{M}$ of $\text{PA}$, we construct a model of $\text{LTT}_0$. The objects of $\mathcal{E}$ are the sets built up from $\mathcal{M}$ by $\times$, $\to$ and $P$. The objects of $\mathcal{B}$ are the sets of all sequences of objects of $\mathcal{E}$. The arrows are the $\text{hor}$ functions. The objects of $\mathcal{P}$ over $b \in \mathcal{B}$ are all subsets of $b$.

Note that $\mathcal{E}$ and $\mathcal{P}$ are radically different.
Conservativity of $\text{LTT}_0$ over PA

I have previously given syntactic proofs that $\text{LTT}_0$ is conservative over PA. We can now give a semantic proof of the same result.

**Theorem**

$LTT_0$ is conservative over PA.

**Proof.**

From any model $\mathcal{M}$ of PA, we construct a model of $\text{LTT}_0$.

The objects of $\mathbb{E}$ are the sets built up from $|\mathcal{M}|$ by $\times$, $\rightarrow$ and $P$.

The objects of $\mathbb{B}$ are the sets of all sequences of objects of $\mathbb{E}$. The arrows are the hor functions.

The objects of $\mathbb{P}$ over $b \in \mathbb{B}$ are all subsets of $b$.

We can similarly prove $\text{LTT}_0$ conservative over ACA$_0$.

**Corollary**

ACA$_0$ is conservative over PA.
Bounded Quantification

Problem: How do we turn $\text{prop}$ into the set of $\Sigma_0$-propositions?
Bounded Quantification

Problem: How do we turn \texttt{prop} into the set of $\Sigma_0$-propositions? Just close it under bounded quantification? Categorical semantics are horrible.

Put (a name of) \texttt{prop} in $U$. We can define bounded quantification by elimination $N$ over \texttt{prop}:

\[
\forall x < 0. \phi(x) = \top \\
\forall x < S(n). \phi(x) = \forall x < n. \phi(x) \land \phi(n)
\]

Conversely, any formula in \texttt{prop} in the new LTT corresponds to a $\Sigma_0$-formula in $I_{\Sigma_0}(\exp)$.

(Show that the functions in $N \to N$ are all defined by a $\Sigma_0$-formula in $I_{\Sigma_0}(\exp)$. Use the fact that the $\Sigma_0$-definable functions are closed under primitive recursion.)
Bounded Quantification

Problem: How do we turn \( \text{prop} \) into the set of \( \Sigma_0 \)-propositions? Just close it under bounded quantification? Categorical semantics are horrible.

Answer: Put (a name of) \( \text{prop} \) in \( U \).
Bounded Quantification

Problem: How do we turn $\text{prop}$ into the set of $\Sigma_0$-propositions? Just close it under bounded quantification? Categorical semantics are horrible.

Answer: Put (a name of) $\text{prop}$ in $U$.

We can define bounded quantification by elimination $\mathbb{N}$ over $\text{prop}$:

$$
\forall x < 0. \phi(x) = \top
$$
$$
\forall x < S(n). \phi(x) = \forall x < n. \phi(x) \land \phi(n)
$$
Bounded Quantification

Problem: How do we turn \( \text{prop} \) into the set of \( \Sigma_0 \)-propositions? Just close it under bounded quantification? Categorical semantics are horrible.

Answer: Put (a name of) \( \text{prop} \) in \( U \).

We can define bounded quantification by elimination \( \mathbb{N} \) over \( \text{prop} \):

\[
\forall x < 0. \phi(x) = \top \\
\forall x < S(n). \phi(x) = \forall x < n. \phi(x) \land \phi(n)
\]

Conversely, any formula in \( \text{prop} \) in the new LTT corresponds to a \( \Sigma_0 \)-formula in \( I\Sigma_0(\text{exp}) \).
Bounded Quantification

Problem: How do we turn prop into the set of $\Sigma_0$-propositions? Just close it under bounded quantification? Categorical semantics are horrible.

Answer: Put (a name of) prop in $U$.

We can define bounded quantification by elimination $\mathbb{N}$ over prop:

$$\forall x < 0. \phi(x) = \top$$
$$\forall x < S(n). \phi(x) = \forall x < n. \phi(x) \land \phi(n)$$

Conversely, any formula in prop in the new LTT corresponds to a $\Sigma_0$-formula in $I\Sigma_0(exp)$.

(Show that the functions in $\mathbb{N} \rightarrow \mathbb{N}$ are all defined by a $\Sigma_0$-formula in $I\Sigma_0(exp)$. Use the fact that the $\Sigma_0$-definable functions are closed under primitive recursion.)
Conclusion

Logic-enriched type theories are the right setting for investigating many foundational questions in type theory, and in orthodox logic.
Conclusion

Logic-enriched type theories are the right setting for investigating many foundational questions in type theory, and in orthodox logic. At the moment, they are a solution looking for a problem.

*I don’t want to carry on finding new proofs of old results.*
Conclusion

Logic-enriched type theories are the right setting for investigating many foundational questions in type theory, and in orthodox logic. At the moment, they are a solution looking for a problem.

_I don’t want to carry on finding new proofs of old results._

Questions I plan to investigate:

- What is the proof-theoretic ordinal of this LTT?
- What is the set of functions definable in this LTT?
- Some logical features work nicely in LTTs ($\Sigma_0$-induction, $\Sigma_1$-induction)
- Some do not ($\Sigma_2$-induction)
- What’s the difference between these?
Conclusion

Logic-enriched type theories are the right setting for investigating many foundational questions in type theory, and in orthodox logic. At the moment, they are a solution looking for a problem.

*I don’t want to carry on finding new proofs of old results.*

Questions I plan to investigate:

- What is the proof-theoretic ordinal of this LTT?
- What is the set of functions definable in this LTT?
- Some logical features work nicely in LTTs ($\Sigma_0$-induction, $\Sigma_1$-induction)
- Some do not ($\Sigma_2$-induction)
- What’s the difference between these?

Please bring me some more.
Syntax of an LTT

$LTT_0$ is a system with:

Judgement forms:

\[
\begin{align*}
\Gamma &\vdash A \text{ type} \\
\Gamma &\vdash M : A \\
\Gamma &\vdash \phi \text{ Prop} \\
\Gamma &\vdash P : \phi
\end{align*}
\]

and associated equality judgements.

Type

\[ A ::= \]

Term

\[ M ::= x \]

Proposition

\[ \phi ::= \]

Proof

\[ P ::= \]
Syntax of an LTT

Let $\text{LTT}_0$ be a system with:
- arrow types
- product types
- natural numbers
- a type universe closed under $\times$ and $\rightarrow$
- classical predicate logic
- a propositional universe
- typed sets

Type

\[ A ::= A \rightarrow A \]

Term

\[ M ::= x \mid \lambda x : A. M \mid MM \]

Proposition

\[ \phi ::= \]

Proof

\[ P ::= \]
Syntax of an LTT

$LTT_0$ is a system with:
- arrow types
- product types

Type
\[ A ::= A \rightarrow A \mid A \times A \]

Term
\[ M ::= x \mid \lambda x : A.M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \]

Proposition
\[ \phi ::= \]

Proof
\[ P ::= \]
Syntax of an LTT

LTT₀ is a system with:
- arrow types
- product types
- natural numbers

Type

\[ A ::= \ A \to A \mid A \times A \mid \mathbb{N} \]

Term

\[ M ::= \ x \mid \lambda x : A.M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid 0 \mid S(M) \mid E_\mathbb{N}(M, M, M, M) \]

Proposition

\[ \phi ::= \]

Proof

\[ P ::= \]
Syntax of an LTT

LTT_0 is a system with:
- arrow types
- product types
- natural numbers
- a type universe closed under \( \mathbb{N} \) and \( \times \)

```
Type  \quad A :::= A \to A | A \times A | \mathbb{N} | U | T(M)
Term  \quad M :::= x | \lambda x : A.M | MM | (M, M) | \pi_1(M) | \pi_2(M) | 0 | S(M) | E_{\mathbb{N}}(M, M, M, M) | \hat{\mathbb{N}} | M \hat{\times} M |
Proposition  \quad \phi :::=
Proof  \quad P :::=
```
Syntax of an LTT

LTT\(_0\) is a system with:

- arrow types
- product types
- natural numbers
- a type universe closed under \(\mathbb{N}\) and \(\times\)
- classical predicate logic

**Type**

\[
T ::= A \rightarrow A \mid A \times A \mid \mathbb{N} \mid U \mid T(M)
\]

**Term**

\[
M ::= x \mid \lambda x : A . M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M) \mid \hat{\mathbb{N}} \mid M \hat{\times} M
\]

**Proposition**

\[
\phi ::= M =_A M \mid \neg \phi \mid \phi \land \phi \mid \phi \lor \phi \mid \phi \rightarrow \phi \mid \forall x : A . \phi \mid \exists x : A . \phi
\]

**Proof**

\[
P ::= \ldots
\]
Syntax of an LTT

LTT₀ is a system with:
- arrow types
- product types
- natural numbers
- a type universe closed under \( \mathbb{N} \) and \( \times \)
- classical predicate logic
- a *propositional universe*

<table>
<thead>
<tr>
<th>Type</th>
<th>( A ) ::= ( A \rightarrow A )</th>
<th>( A \times A )</th>
<th>( \mathbb{N} )</th>
<th>( U )</th>
<th>( T(M) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term</td>
<td>( M ) ::= ( x )</td>
<td>( \lambda x : A. M )</td>
<td>( MM )</td>
<td>( (M, M) )</td>
<td>( \pi_1(M) )</td>
</tr>
<tr>
<td>Proposition</td>
<td>( \phi ) ::= ( M =_A M )</td>
<td>( \neg \phi )</td>
<td>( \phi \land \phi )</td>
<td>( \phi \lor \phi )</td>
<td>( \phi \rightarrow \phi )</td>
</tr>
<tr>
<td>Proof</td>
<td>( P ) ::= ( \cdots )</td>
<td>( M \hat{=} M )</td>
<td>( \hat{\neg} \phi )</td>
<td>( \phi \hat{\lor} \phi )</td>
<td>( \cdots )</td>
</tr>
</tbody>
</table>
Syntax of an LTT

LTT_0 is a system with:

- arrow types
- product types
- natural numbers
- a type universe closed under \( \mathbb{N} \) and \( \times \)
- classical predicate logic
- a propositional universe
- typed sets

Type

\[ A ::= A \to A \mid A \times A \mid \mathbb{N} \mid U \mid T(M) \mid \text{Set}(A) \]

Term

\[ M ::= x \mid \lambda x : A.M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M) \mid \hat{\mathbb{N}} \mid M\hat{\times}M \mid \{ x : A \mid P \} \]

Proposition

\[ \phi ::= M =_A M \mid \neg \phi \mid \phi \land \phi \mid \phi \lor \phi \mid \phi \to \phi \mid \forall x : A. \phi \mid \exists x : A. \phi \mid \text{prop} \mid V(P) \]

Proof

\[ P ::= \cdots \mid M \hat{=} M \mid \hat{\phi} \mid \phi \hat{\phi} \mid \cdots \mid M \in M \]
We can give a semantic proof of this result:
A function \( f : \mathbb{N}^n \to \mathbb{N} \) is *definable* in PA iff there is a formula \( \phi[x_1, \ldots, x_n, y] \) such that:

1. for all \( a_1, \ldots, a_n \), \( \text{PA} \vdash \phi[\overline{a_1}, \ldots, \overline{a_n}, f(a_1, \ldots, a_n)] \);
2. \( \text{PA} \vdash \forall x_1 \cdots \forall x_n \exists! y \phi[x_1, \ldots, x_n, y] \)

**Theorem**

*The functions definable in PA are exactly the \( \epsilon_0 \)-recursive functions.*

**Proof.**

Construct a model of LTT\(_0\) in which the arrows are the \( \epsilon_0 \)-recursive functions. Then apply conservativity.
History of LTTs

2002 Aczel and Gambino define translations between Constructive ZF (CZF) and the type theory $ML_1 V$. 

2006 Gambino and Aczel introduce the logic-enriched type theory $ML(CZF)$ as a half-way stage.

2007 Adams and Luo show how an LTT can represent Weyl’s school of predicativism.

2010 Adams and Luo show their system is not conservative over PA.
History of LTTs

2002 Aczel and Gambino [?] define translations between Constructive ZF (CZF) and the type theory $\text{ML}_1 V$.

2006 Gambino and Aczel [?] introduce the logic-enriched type theory $\text{ML(CZF)}$ as a half-way stage.
History of LTTs

2002 Aczel and Gambino define translations between Constructive ZF (CZF) and the type theory $ML_1 V$.

2006 Gambino and Aczel introduce the logic-enriched type theory $ML(CZF)$ as a half-way stage.

2007 Adams and Luo show how an LTT can represent Weyl’s school of predicativism.
History of LTTs

2002  Aczel and Gambino [?] define translations between Constructive ZF (CZF) and the type theory $\text{ML}_1 \text{V}$.

2006  Gambino and Aczel [?] introduce the logic-enriched type theory $\text{ML}(\text{CZF})$ as a half-way stage.

2007  Adams and Luo [?] show how an LTT can represent Weyl’s school of predicativism.

2010  Adams and Luo [?] show their system is not conservative over PA.
The Moral of the Story

From this work, I take the message:

- LTTs can do \textit{some} things better than either orthodox logics or type theories.
The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- ... which becomes just one of a family of translations.
The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- . . . which becomes just one of a family of translations.
- But I need semantics to guide future research.
The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- ...which becomes just one of a family of translations.
- But I need semantics to guide future research.
- I also need to think of better names.
The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- ...which becomes just one of a family of translations.
- But I need semantics to guide future research.
- I also need to think of better names.
The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- ... which becomes just one of a family of translations.
- But I need semantics to guide future research.
- I also need to think of better names.

I do *not* claim:

- LTTs are ‘better’ than predicate logics or type theories.
The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- ...which becomes just one of a family of translations.
- But I need semantics to guide future research.
- I also need to think of better names.

I do *not* claim:

- LTTs are ‘better’ than predicate logics or type theories.
- Curry-Howard is ‘bad’